基于 XML 及反射技术的多语言界面研究与实现

顾薛平,吴 宁
(南京航空航天大学 电子信息工程学院,江苏 南京 210016)

摘 要:针对飞行员飞行品质评估系统中的多语言界面问题,采用扩展 XML 外部语言包的方法,结合反射技术,实现了 C# 语言环境下对应用软件的多语言支持。实现过程中对界面文字的导入/导出功能进行了封装处理,消除了冗余代码。该方法对源程序修改较少,而扩展性强、维护成本小、程序直观性好。

关键词: XML;反射技术;多语言界面;C#
中图分类号: TP319 **文献标识码**: A **文章编号**: 1674-7720(2012)11-0001-03

Research and implementation of multi-language interface based on XML and reflection technology

Gu Xueping, Wu Ning
(College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract: Aiming to solve the multi-language interface problem in the pilot performance assessment tools, the external XML language package is adopted as well as reflection technology and the multi-language support for the software in C# is finally realized in this paper. Besides, the language import and export function is encapsulated to eliminate redundant code. The method has a good visibility and expansibility, economizes cost of maintenance and leads to the program a good visual effect.

Key words: XML; reflection technology; multi-language interface; C#

为保证飞行安全、降低事故发生率,国内外民航系统普遍重视飞行员的培训工作。目前,国内飞行员培训中心对飞行员操作水平的评估一般都是采用教练员观察和打分的方式,不够客观和全面。飞行员飞行品质评估系统是一个能够实现自动规范评分的软件,它面向的客户是训练中心的教员、各航空公司的受训飞行员和高层管理者等,因此,要求软件能够针对不同国家的用户,提供不同语言版本的界面,以便于用户能够更快地接受和使用该产品。

多语言界面软件的设计,通常采取应用程序和界面文字显示相分离的技术[1]:程序代码独立编写,语言资源从核心代码中分离出来储存到文件,运行时根据用户需求选择相应的语言资源文件并显示。常用的方法有:(1)将系统支持的语言资源生成一个多语言的可执行文件或独立的资源文件[2]。这种方法比较常用,但是资源更新时需要重新编译,扩展性差;(2)将支持的语言资源存放在外部语言包中,通过替换语言包达到切换语种的目的[3-4]。这种方法不需要重新编译程序,外部语言包保持独立,可以动态修改。

基于软件开发过程中提出的对源程序进行最少的修改,达到尽可能高的效率、可靠性和扩展性的要求,本文利用 XML 外部语言包实现多语言界面,并且在系统运行的同时生成语言包文件,便于扩展;对界面文字的导入/导出功能进行了封装处理,避免了代码的冗余,对源程序修改代价最小,后期维护方便;在语言包处理过程中运用了.NET 的反射技术以提取和组装界面提示文字信息,增加了程序的灵活性和直观性。

1 XML 语言文件包格式

实现多语言界面的一个重要步骤是设计外部语言包文件。可扩展标记语言 XML 具有自解释性和灵活的结构,可实现跨平台交互,表述任意复杂程度的数据格式和无限量的自定义格式描述符,适合作为语言包的格式。

用户从软件获取的信息都显示在用户界面上,用户

界面由控件对象组成，而语言信息都是由文本控件对象的属性值来体现的。建立 XML 语言包文件的基本思想，就是从用户界面中获取所有文本控件的标识和属性值，按照结构类别存放到 XML 文档中，这样语言包文件即可通过控件的标识映射到用户界面。应用程序的用户界面通常包含的文本组件如图 1 所示。

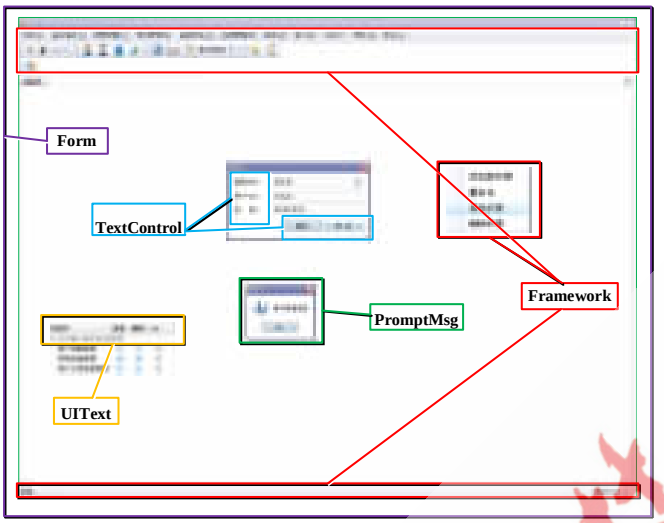

图 1 应用程序界面示意图

用户界面要处理的文本控件对象主要有以下部分：

(1) Framework：菜单栏、工具栏、状态栏等显示文本信息的窗体框架，其文字信息包括各栏框架 Bar 的 Text 属性、各栏菜单项 baritem 的 Caption 属性（显示文字）和 Hint 属性（提示文字）。基于美化界面和方便使用的要求，本文中使用 BarManager 控件来组装 Framework，在程序中分别遍历其 Bar 和 Items 成员即可获取框架和菜单项的信息。

(2) TextControl：只具有（需要）显示文字功能的控件，如标签控件 Label 等。

(3) PromptMsg：人机交互过程中显示的提示信息。提示信息是解决方案中所有窗体共用的，因此，将其以公共静态变量的形式存放在一个全局静态类中。软件运行时，利用 C# 的反射技术动态获取该类的信息，或创建类的实例并调用和访问这些实例。

(4) UIText：在程序中需要处理而在界面上没有显示（或需要经过处理后显示）的文字信息，如表格控件 GridView 的表头等。

为确保切换语言时能够置换所有的界面文字信息，所有的文本控件对象都必须在 XML 语言包中有唯一的节点与之对应。本文中设定的 XML 语言包文件的结构如图 2 所示。

根据文本控件在软件界面中的关系，采用树形结构来描述语言环境，以解决方案为根节点；所有的窗体都是解决方案的子节点，其子节点是窗体中各种结构类别的文本控件；解决方案根节点的子节点还包括全局提示信息，其子节点是所有的全局提示信息。

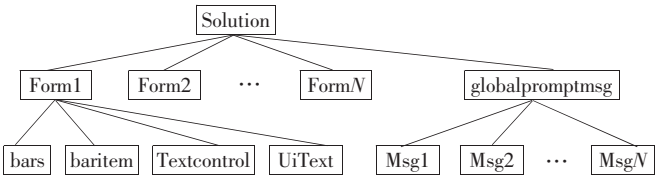

图 2 XML 语言包文件的树形结构示意图

2 多语言界面的实现

2.1 软件模块与数据结构设计

程序设计中采用独立界面独立模块的模式，每个模块均由一个独立界面提供相应功能。.NET 中定义 Form 类为所有窗体类的基类，项目中窗体间的关系如图 3 所示。

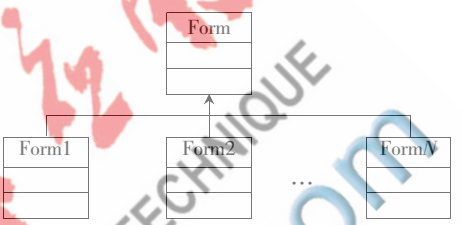

图 3 Form 窗体模块图通常形式

从图 3 中可看出，项目中所有的窗体类均继承自 Form 类。考虑到导入与导出语言信息是每个模块界面都应具有的功能，如果在每个窗体中都添加实现语言导入/导出的代码，会造成代码的冗余，不易维护，也不是一个良好的设计。因此，本文在 Form 与其他窗体之间增加一层，提供界面语言的导入与导出，如图 4 所示。

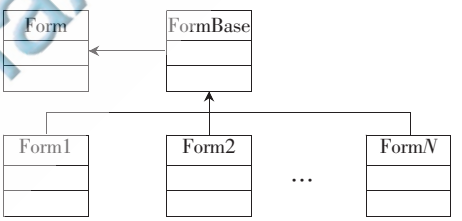

图 4 本文中的 Form 窗体模块图

FormBase 继承自 Form 类，其余所有窗体以 FormBase 为父类。本文将界面文字信息的导入和导出功能封装在基类 FormBase 的 Load 事件中。派生窗体类与基类之间通过特定的数据结构来传递数据：链表 TextControl 用于存放 Label 等需显示文本的控件；隐含在程序中的文字信息存入数组 UIText 中，根据实际情况进行处理；程序提示信息以静态变量的形式放在全局静态类 Globalpromptmsg 中，以提供全局的提示信息管理机制、简化语言信息的提取与替换。程序运行时导入和导出文本信息的执行流程如图 5 所示。

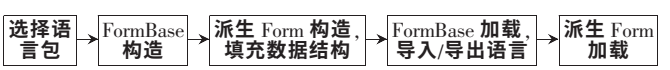

图 5 界面文字信息导入导出流程图

程序运行前选择需要的语言包文件。运行后每个窗体被打开时，首先在构造函数中完成必要的数据结构填

充，继而在加载基类时实现语言信息的导出与载入，最后加载窗体。只要每个派生的 Form 类填充了相应的数据结构，即可以利用基类中的语言信息导出与载入功能，实现整个软件所有窗体的语言替换。

2.2 界面文字信息的导出

对于大型的工程项目，窗体数目多、组件复杂，手动建立 XML 语言包文档不仅耗费大量时间，而且容易出错。因此，本文在程序运行时生成 XML 文档，每打开一个窗体都将其界面文字信息导入到 XML 语言包文档。反射是.NET 中的重要机制，它使得.NET 中的类型（包括类、结构、委托和枚举等）可以通过名称动态检索成员信息，并且允许在运行的程序中操作这些信息。本文运用反射技术动态地提取界面提示文字信息，具体的实现方法如下：

（1）程序启动时，创建 XML 文档，写入解决方案根节点。

（2）打开一个窗体，在窗体的构造函数中完成该窗体的数据结构填充，加载时完成文字信息的导入。

（3）若当前处理的窗体未被处理过，在 XML 文档根节点下创建一个新的 Form 子节点，写入窗体的名称和标题栏文本等属性，并标记处理过的窗体以防重复处理。对于窗体的 Framework 控件，在 Form 节点下创建 bars 节点和 baritem 节点，将所有的框架和菜单项的名称和文本以 item 节点的形式分别添加到这两个节点下；对于 TextControl 类控件，在 Form 节点下创建 textcontrol 节点，将所有该类控件的名称和文本以 item 节点的形式添加到 textcontrol 节点下；对于 UIText 类组件，在 Form 节点下创建 uitext 节点，将该类组件的名称和文本以 item 节点的形式添加到 uitext 节点下。

（4）全部窗体处理结束（即退出程序时），在根节点下创建 globalpromptmsg 子节点，利用反射机制获取全局静态类 Globalpromptmsg 声明的所有字段信息，将所有字段的名称和文本写到 globalpromptmsg 节点下 item 节点的 name 属性和 text 属性。全部字段处理结束后向 XML 文档中写入结束标志，完成界面文字信息的导出。其程序如下：

```
Type type = Globalpromptmsg.GetType();
FieldInfo [] currfiledinfos =type.GetFields (BindingFlags.
        Public|BindingFlags.Static|BindingFlags.Instance);
foreach (FieldInfo filedinfo in currfiledinfos)
{
  XmlWriter.WriteStartElement("item");
  XmlWriter.WriteAttributeString("name",filedinfo.Name);
   XmlWriter.WriteAttributeString ("text",filedinfo.GetValue
                                    (filedinfo).ToString());
     XmlWriter.WriteEndElement();
    }
```

字段是在类中定义的变量，字段信息从元数据中获取，FieldInfo 类发现字段属性并提供对字段元数据的访问权。FieldInfo 类没有公共构造函数，故先用 GetType 方法获取全局静态类的 Type 对象（关于类型声明的信息），然后调用 Type 对象的 GetFields 方法来获取 FieldInfo 对象。FieldInfo 对象通过 GetValue 方法返回给定字段的值。

系统支持的语言扩展时，只要将生成的语言包文档中各文本控件节点的 text 属性值翻译成目标语言，即可得到需要的语言包文件。

2.3 界面文字信息的导入

在创建了语言包文件的基础上，如何将程序和语言包文件结合起来，成为实现多语言界面的最后一个难题。本文在程序启动时读取选择的 XML 语言包文件，将数据记录到内存，窗体加载时进行数据填充，替换默认的界面文字。导入的过程中也使用了反射技术，其方法如下：

（1）程序启动时，打开要读取的外部 XML 语言包文件。

（2）按深度遍历方法读取 XML 文件。提取所有 Form 节点的名称和标题栏文本属性值，存入内存。每个 Form 创建一个窗体信息对象，存放 Framework、TextControl 以及 UIText 组件的文本信息，多个窗体信息对象之间以标题栏文本为区分标识。

（3）所有窗体处理完毕后，替换全局提示信息文本：创建 Globalpromptmsg 类的一个实例，获取它的类型信息，搜索该实例中与 globalpromptmsg 节点下提示信息节点同名的公开字段，利用反射机制将对应字段的值替换为对应节点的 text 属性值。其程序如下：

```
Globalpromptmsg gmp=new Globalpromptmsg ();
Type type=gmp.GetType();
FieldInfo currfieldinfo = null;
foreach(XmlNode msgnode in formnode.ChildNodes)
{
 currfieldinfo=type.GetField (msgnode.Attributes["name"].
                                          Value);
 if (currfieldinfo ! = null)
    currfieldinfo.SetValue(currfieldinfo, msgnode.
                          Attributes["text"].Value);
}
```

通过调用 Type 对象的 GetField 方法来搜索具有指定名称的公开字段，并将其存入 FieldInfo 对象；然后调用 FieldInfo 对象的 SetValue 方法对给定字段值进行修改。

（4）窗体加载时，从第 2 步获取的数据中查找对应控件的各项文本属性信息，替换掉组件默认的文本，从而实现界面语言的导入。

2.4 实际界面显示

以登录界面为例，提取出一个简单的语言为英文的 xml 文件如图 6 所示。

图 6 登录界面的 xml 语言包文件

图中，Solution 节点代表解决方案，Native-lan 节点代表显示的语言种类，Form 节点代表窗体，Form 的子节点 textcontrol 和 uitext 分别对应窗体需显示文本的控件和隐含在程序中的文字信息。提示信息存放在 globalpromptmsg 节点下。每个 item 节点都指明了组件的名称和显示文本，扩展支持语言时只需修改节点的 Text 值。

本设计中默认语言是中文简体，默认的登录界面如图 7 所示。选择英文语言包文件时，程序运行得到的登录界面如图 8 所示。

本文采用扩展外部 XML 语言包的方法，在对源程序修改代价最小的情况下实现了对软件的多语言支持，语言增加或界面变化时无需重新编译源程序，动态增加或修改语言包文件即可，实现简单、扩展性强、维护成本小，是一个设计良好的多语言资源实现模型。在界面语言导入导出时使用了.NET 反射技术、动态调用需要的方法和属性信息，其程序灵活直观。本文方法在飞行员飞行品质评估系统中得到了成功应用。

图 7 中文界面

图 8 英文界面

参考文献

[1] 陈传波，洪慧芳.基于 XML 的本地化技术研究[J].计算机工程与科学，2006，28(10)：95-97.
[2] 司国东. .NET 环境下的一种多语言界面解决方案[J].农业网络信息，2007(2)：35-36.
[3] 唐勇，李秀龙.多语言用户界面的研究与实现[J].计算机应用研究，2002(4)：112-113.
[4] 王锋，魏晓丽，江开耀，等.基于 XML 的 C# 多语言界面实现[J].计算机工程与设计，2008，29(15)：4073-4078.

(收稿日期：2012-02-08)

作者简介：

顾薛平，女，1988 年生，硕士，主要研究方向：数字系统设计与计算机应用，边界扫描测试与可测性设计，FPGA 系统设计。